Int J CARS (2015) 10:1873–1891
DOI 10.1007/s11548-014-1142-5

ORIGINAL ARTICLE

Detection and modelling of contacts in explicit finite-element simulation of soft tissue biomechanics

S. F. Johnsen · Z. A. Taylor · L. Han · Y. Hu · M. J. Clarkson · D. J. Hawkes · S. Ourselin

Received: 27 September 2014 / Accepted: 16 December 2014 / Published online: 6 January 2015

Abstract
Purpose Realistic modelling of soft tissue biomechanics and mechanical interactions between tissues is an important part of biomechanically-informed surgical image-guidance and surgical simulation. This submission details a contact-modelling pipeline suitable for implementation in explicit matrix-free FEM solvers. While these FEM algorithms have been shown to be very suitable for simulation of soft tissue biomechanics and successfully used in a number of image--guidance systems, contact modelling specifically for these solvers is rarely addressed, partly because the typically large number of time steps required with this class of FEM solvers has led to a perception of them being a poor choice for simulations requiring complex contact modelling.
Methods The presented algorithm is capable of handling most scenarios typically encountered in image-guidance. The contact forces are computed with an evolution of the Lagrange-multiplier method first used by Taylor and Flanagan in PRONTO 3D extended with spatio-temporal smoothing heuristics for improved stability and edge–edge collision handling, and a new friction model. For contact search, a bounding-volume hierarchy (BVH) is employed, which is capable of identifying self-collisions by means of the surface-normal bounding cone of Volino and Magnenat-Thalmann, in turn computed with a novel formula. The BVH is further optimised for the small time steps by reducing the number of bounding-volume refittings between iterations through identification of regions with mostly rigid motion and negligible deformation. Further optimisation is achieved by integrating the self-collision criterion in the BVH creation and updating algorithms.
Results The effectiveness of the algorithm is demonstrated on a number of artificial test cases and meshes derived from medical image data. It is shown that the proposed algorithm reduces the cost of BVH refitting to the point where it becomes a negligible part of the overall computation time of the simulation. It is also shown that the proposed surface-normal cone computation formula leads to about 40 % fewer BVH subtrees that must be checked for self-collisions compared with the widely used method of Provot. The proposed contact-force formulation and friction model are evaluated on artificial test cases that allow for a comparison with a ground truth. The quality of the proposed contact forces is assessed in terms of trajectories and energy conservation; a <0.4 % drop off in total energy and highly plausible trajectories are found in the experiments. The friction model is evaluated through a benchmark problem with an analytical solution and a maximum displacement error of 8.2 %, and excellent agreement in terms of the stick/slip boundary is found. Finally, we show with realistic image-guidance examples that the entire contact-modelling pipeline can be executed within a timeframe that is of the same order of magnitude as that required for standard FEM computations.

Keywords Contact modelling · Collision detection · FEM · Total Lagrangian explicit dynamics · Soft tissue biomechanics

S. F. Johnsen (✉) · Y. Hu · M. J. Clarkson · D. J. Hawkes · S. Ourselin
Centre for Medical Image Computing, University College London, London, UK
e-mail: rmapsfj@live.ucl.ac.uk

Z. A. Taylor
Department of Mechanical Engineering, CISTIB Centre for Computational Imaging and Simulation Technologies in Biomedicine, Insigneo Institute for in Silico Medicine, The University of Sheffield, Sheffield, UK

L. Han
School of Medicine, Tongji University, Shanghai, People's Republic of China

Introduction

FEM modelling has for some time now played an important role in surgical simulation [1,32] and is finding its way into surgical guidance [3,5]. Explicit FEM solvers, particularly such based on the total Lagrangian explicit dynamics (TLED), have been shown to provide a versatile and realistic means of simulating soft tissue solid dynamics, which is at the core of such guidance systems [19,26,32]. Their decoupling of the degrees of freedom also makes them ideal candidates for parallelisation, which in recent years with the advent of general-purpose GPUs and multi-core mainstream CPUs has proved to be a great source of cost-efficient execution speed [34,35].

They do, however, suffer from the inherent shortcoming of only allowing for small time steps, which can make simulations involving large-deformation contact modelling prohibitively expensive mainly due to the costs associated with contact search. Another drawback is that, compared with implicit methods, little literature is available on contact modelling for these solvers. The commonly encountered ones are the penalty force and the Lagrange-multiplier method of Taylor and Flanagan, and Heinstein et al. [14], among the force-based methods [37], and kinematic contacts that rely on a direct correction of displacements and are very efficient, but are only capable of modelling contacts between deformable and rigid bodies [11]. All of these are typically implemented as node-segment contact algorithms only capable of detecting penetration of mesh nodes into surfaces, which requires two detection passes to achieve some degree of separation of the two surfaces in contact, and even with those two passes, mesh edges are still free to intersect. Node-segment methods must rely on denser meshes to avoid the latter type of mesh intersection which in turn entails more and computationally costlier time steps.

The algorithm presented in this paper was implemented as the general- purpose contact-modelling component of the open-source[1] TLED-based FEM solver package *NiftySim* [19]. It attempts to carry over some of the developments made in the context of implicit contact-modelling algorithms to explicit methods, such as being able to process contacts in a single pass as with segment–segment methods [31], provided the meshes in contact have a similar resolution. Spatial smoothing, which attempts to alleviate the stability issues caused by sudden changes in the direction of contact forces arising from the use of coarse, piece-wise linear contact surfaces has found widespread adoption in implicit methods [30,38], is also employed. Further stability improvement is achieved by gradually slowing down approaching contact surfaces in close proximity, thus adding temporal smoothing to the method.

[1] http://niftysim.sourceforge.net.

Another major area of focus in this work is the reduction of contact search costs through bounding-volume hierarchies (BVH) with novel, time-saving update and self-collision detection heuristics. For self-collision detection, we employ the surface-normal bounding-cone heuristics developed by Volino and Magnenat-Thalmann [36]. New formulas for the computation of the bounding cones, via Provot's recursive algorithm [29], are introduced. Another novel aspect is how the self-collision criterion is deeply integrated in determining the topology of the BVH and the decision on when to update BVH subtrees. The BVH updating algorithm is specialised for the typically small time steps of explicit methods, in which it comprises a method for the characterisation of the deformation the simulation geometry has undergone and identification of areas of negligible deformation and rigid motion, and updating of the BVH of the latter parts by means of rigid geometrical transforms.

The proposed method is further notable due to its versatility; it allows for modelling of contacts between the surfaces of two solid meshes, self-collisions, contacts between solid and membrane meshes, and deformable bodies interacting with moving or fixed rigid ones, and a new, simple friction model is available, too.

This paper is organised as follows: After a brief overview of related previous work (section "Related work"), section "Total Lagrangian explicit dynamics" contains an introduction of the underlying FEM algorithm, the total Lagrangian explicit dynamics. This is followed by a detailed discussion of the contact-modelling pipeline that is subdivided into a relatively short part describing the contact surface data structures (section "Contact surfaces") and two larger sub-sections, the first of which deals with the contact search (section "Contact search"). Novel modifications to the self-collision detection method and the new BVH creation and update strategies are discussed in this order, in this part of the paper. The contact model is developed in section "Contact-force calculations"; it starts with a discussion of penetration response forces for node–facet and edge–edge collisions, followed by a discussion of the temporal smoothing and the friction model. In section "Experiments", the BVH algorithms are validated in the order in which they are presented by comparison with some alternatives that swap out some of the novel aspects for simpler or more established methods, on mostly synthetic test cases. The contact model is validated in terms of energy and momentum conservation, and plausibility of the trajectories resulting from impacts (section "Contact forces"). In section "Friction", the friction model is validated on a benchmark problem with an analytical ground truth taken from the literature. Finally, a demonstration of the entire pipeline's performance on two image-guidance problems is provided (section "Examples from image-guidance").

Related work

Classically, the algorithms for FEM contact modelling are node-segment approaches [37], where one of the two surfaces in contact is assigned the role of the *slave* surface, the other is called the *master* surface. The only type of mesh inter-penetration node-segment approaches can resolve are those of the master surface by slave nodes, which in turn necessitates two contact search and resolution steps with alternating master-slave roles for every time step. The underlying principle of mesh intersection handling with node-segment methods is to project slave nodes onto the nearest facet of the master surface and check the sign of the difference between the slave node position and its projection with respect to the master surface normal. A contact response in direction of the master surface normal is then applied. Since most FEM elements are only C0 continuous, this approach can also lead to sudden jumps in the direction of the response experienced by a node sliding over the master surface, in turn leading to instability of the algorithm. Smoothing node-segment methods, such as the one by Wriggers and Krstulović-Opara [38] based on cubic Bézier polynomials, were introduced to remedy this issue.

Segment–segment contact algorithms were devised to overcome the need for two passes with the node-segment approach [31]. Mortar elements, originally developed for coupling non-conforming meshes, were introduced to the field of contact modelling to also overcome the various mathematical limitations of the early node-segment methods, mainly stability problems with implicit methods arising from non-satisfaction of the Babuska-Brezzi condition. In these methods, the contacting meshes are pushed apart by a contact pressure that is interpolated over the mortar mesh. The work of Puso and Laurensen [30] introduced a mortar method for 3D and large deformations.

An interesting method suitable for any FEM or similar algorithm that assembles stiffness matrices was developed by Duriez et al. [7]. Their contact model based on Signorini's law was primarily designed with haptics in mind and computes contact forces from the constitutive model of the bodies in contact.

An alternative to both node–segment and segment–segment methods, based on intersection volumes was devised by Heidelberger et al. [13] and significantly extended by Allard et al. [2]. By employing layered depth images (LDI) for intersection volume computation, they effectively solved collision detection and response calculation using the same method. However, the method is limited in its application to volumetric meshes.

A notable development in the area of matrix-free explicit FEM algorithms came with the Lagrange-multiplier-based method employed in PRONTO 3D by Taylor and Flanagan [33], and later extended by Heinstein et al. [14]. These methods, like the—probably most widely adopted for explicit FEM— penalty method, resolve mesh intersection by applying forces to the offending nodes. Unlike with penalty methods that contain an arbitrary non-physical parameter, the forces with the Lagrange- multiplier method arise directly from the discretised equilibrium equation and lead to an immediate resolution of any mesh intersection and do not affect the admissible time-step size [4]. Cirak and West [6] devised a method for simulating impact contacts with explicit solvers, based on an elaborate decomposition of contact responses into mesh inter-penetration responses and momentum exchanges. In terms of scope, their work resembles ours strongly, with their ability to simulate membrane and solid mesh contacts, frictional as well as frictionless contacts, and handling of both edge–edge and node–facet collisions. Their resolution of mesh inter-penetration was based on a unilateral projection of slave nodes onto the master surface, the consequences of which on the energy balance of the simulated system they tried to minimise by establishing an equation system incorporating both penetration responses and momenta.

The range of contact search algorithms proposed for FEM contact modelling is as wide as that of methods for their solution. We propose a bounding-volume hierarchy (BVH)-based method. These methods are very versatile and used in a wide range of applications such as cloth modelling [25], robot motion planning [10], ray tracing [20], and FEM contact modelling [28,39]. What makes them interesting for the application with the relatively small time steps required with explicit FEM solvers is the ability to take a more localised, selective approach to collision detection and exploitation of temporal coherence. A further advantage of employing a BVH is that it can also be used for other problems arising in surgical image guidance, such as fast point location for point-set registration purposes.

Early developments in the field of BVHs were limited to rigid or even static problems, but since the 1990s, there has been a growing interest in collision detection for simulation of deformable bodies [24]. A key development came in 1994 with Volino and Magnenat-Thalmann's method [36] for efficient self-collision detection. They established two conditions under which a piece of simulated cloth could self-intersect: either the surface is folded onto itself, i.e. it has surface normals pointing in opposite directions, or there are intersecting boundary edges. Larsson and Akenine-Möller [21] devised a hybrid bottom-up/top-down BVH strategy for detecting collisions between deformable bodies. Its purpose is to reduce the number of bounding volume (BV) updates by only updating down to leaf level those parts of the bodies' BVHs that overlap. The method, however, was not adapted for self-collision detection. They later [22] described a method for dynamically creating BVHs for triangle soups particularly suitable for such resulting from fracturing of objects. They also developed a variant of their algorithm

incorporating a sweep and prune sort of all simulation primitives suitable for detecting self-collisions.

Methods

Total Lagrangian explicit dynamics

The TLED class of FEM solvers are matrix-free solvers relying on explicit central-difference time integration. They have enjoyed some success in the simulation of soft tissue biomechanics thanks to their ability to simulate large deformations, the relative ease with which complex material models can be implemented, and not least the possibility for very elegant parallel implementations [26,34,35].

The discretised equilibrium equations of TLED, neglecting damping terms, read:

$$\frac{1}{\Delta t^2} M \left(U^{(t+\Delta t)} - 2U^{(t)} + U^{(t-\Delta t)} \right) + F(U^{(t)}) = R^{(t)} \tag{1}$$

where $U^{(t+\Delta t)}$, $U^{(t)}$, $U^{(t-\Delta t)}$ denote the next, current, and previous time-step displacements, respectively, Δt is the time-step size, R is the external load vector, and M is the lumped (diagonal) mass matrix. The term $F(U)$ represents the internal forces of the current configuration. The evaluation of the latter term does not involve the assembly of a stiffness matrix, instead internal forces are computed directly per element and subsequently accumulated for all nodes. In this work, internal forces are modelled with the neo-Hookean material model, whose strain-energy density function is given by

$$W = \frac{G}{2} \left(\overline{I}_1 - 3 \right) + \frac{K}{2} (J - 1)^2 \tag{2}$$

where G and K denote the shear and bulk modulus of the material, respectively, and $\overline{I}_1 = J^{-2/3}(C_{11} + C_{22} + C_{33})$ with C denoting the left Cauchy-Green deformation tensor and J is the determinant of the deformation gradient. Shell-element internal forces are computed with the EBST shell triangle of Flores and Oñate [8].

Contact algorithm overview

The algorithm is of a predictor-corrector type that first evolves the displacements with the standard TLED algorithm without any regard to contacts, then identifies intersecting or very close geometry, and applies forces to correct the situation. The contact modelling pipeline comprises three major groups of routines and data structures: the *contact surfaces* which contain the geometry that is searched for collisions and some additional data required for contact-force application, the *BVHs* employed in contact search, and finally the *contact-force computation* algorithms.

A pseudo-code overview of the algorithm including the TLED-related computations is given in "Appendix A".

Contact surfaces

Contact surfaces are data structures central to the contact modelling algorithm that consist of the geometric primitives —triangles are employed in the subsequent experiments and some of the explanations—which are tested for collisions and provide extended geometric information required in contact search such as surface normals and projection operators.

The contact surfaces associated with fixed rigid geometry are static data structures for which all normals, projection operators required for contact search and force calculation are precomputed. Moving rigid contact surface data structures are identical to their spatially fixed counterparts apart from possessing an update routine that applies the appropriate translation and/or rotation to the precomputed normals and operators.

The most important type is the deformable contact surface obtained by extracting the surface facets from the simulation solid mesh. Lazy evaluation with caches is employed for normals and other contact search-related data such as projection operators, which allows the algorithm to limit their re-computation to regions that are in contact with or close proximity to other geometry. Contact forces which are calculated for a contact surface node are applied to the corresponding FEM node via an index lookup table that is constructed together with the surface mesh.

If the simulation contains membranes, these elements are included in the same contact surface object as the solid mesh surface facets. To account for the thickness of the membrane, two contact primitives are introduced for every membrane element, one for the top and one for the bottom. The nodes associated with these membrane contact primitives are obtained by offsetting the membrane nodes by half the thickness of the membrane in direction of the normal and its opposite, for top and bottom, respectively, yielding the sandwich structure visible in Fig. 13. By having the entries in the contact-force index lookup table point to the same FEM membrane node for the top and bottom node, it is ensured that contact forces are correctly incorporated in the global force vector.

Contact search

In the following, the algorithm is explained for BVHs that have a binary-tree structure and axis-aligned bounding boxes (AABB) are used for illustrations. However, the presented methods are not limited to this BVH-type.

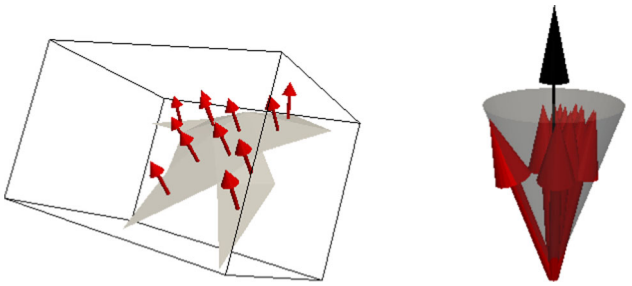


Fig. 1 Surface-normal bounding cones for self-collision detection. *Left*: a patch of connected geometry primitives defining a cone with the corresponding surface normals (*red*) and its AABB. *Right*: the corresponding cone, the normals it bounds (*red*) and the cone axis (*black*)

At leaf level, the BVs bound one primitive each such that the primitive's vertices at the start of the time step as well as at the end of the predictor step are fully contained within it. The leaf BVs are also fitted with a safety margin ϵ_{BV}, which is uniform throughout the BVH and defaults to $\frac{1}{100}(h_{\max} + h_{\min})$ in our implementation, with $h_{\max}$, $h_{\min}$ being the maximum and minimum initial-configuration surface facet diameters. The purpose of this margin is to allow for some geometry deformation without the need to refit the bounding volume and to allow for the detection of primitives in close proximity.

All deformable geometry is contained in one BVH, rigid contact surfaces, moving or fixed, each have each their own BVH.

Self-collisions and surface-normal bounding cones

The surface-normal bounding cones (NBC) are a means for identifying BVH subtrees containing connected geometry that is folded onto itself, i.e. has normals pointing in opposite directions, and thus potentially self-colliding [36], and are illustrated in Fig. 1. Since we mostly deal with solid elements, self-collisions resulting from intersecting mesh boundaries as described by Mezger [25] are not considered, and we treat the NBC self-collision criterion

$$\alpha_{\mathrm{VMT}} \geq \tau_{\mathrm{VMT}}, \quad \tau_{\mathrm{VMT}} \leq \pi \tag{3}$$

where α_{VMT} is the cone opening angle and τ_{VMT} the threshold above which self-collision tests are performed, as a necessary criterion for self-collision. The computation of this quantity α_{VMT} is done recursively as part of the BVH update with a method similar to that proposed by Provot [29], starting with the facet normal of the bounded primitive and an opening angle $\alpha_{\mathrm{VMT}} = 0$, at leaf level, and creating NBCs for interior nodes by merging the cones of their children. Unlike Provot's algorithm that adopts for the parent cone's axis, the average of the child cones' axes and then computes the opening angles such that the child cones are fully contained, our approach computes the narrowest possible NBC from both the child cones' axes ($\boldsymbol{a_1}$ and $\boldsymbol{a_2}$) and opening angles (α_1 and α_2), as illustrated in Fig. 2.

The computation of the parent axis is accomplished via two weights $w_1 > 0,\ w_2 > 0$:

$$\begin{aligned} e &:= \boldsymbol{a_1}^{\mathrm{T}}\boldsymbol{a_2} \\ \beta_1 &= (2\arccos(e) - (\alpha_1 - \alpha_2))/4, \quad \beta_2 = \beta - \beta_1 \\ w_1 &= \cos(\beta_1) - e\frac{\cos(\beta_2) - e\cos(\beta_1)}{1-e^2}, \quad w_2 = \frac{\cos(\beta_2) - e\cos(\beta_1)}{1-e^2} \\ \boldsymbol{a_p} &= \frac{1}{||w_1\boldsymbol{a_1} + w_2\boldsymbol{a_2}||}(w_1\boldsymbol{a_1} + w_2\boldsymbol{a_2}) \end{aligned} \tag{4}$$

A step-by-step derivation of this formula can be found in "Appendix B".

The algorithm only performs these calculations after first checking whether one of the child cones is fully contained in the other. If so, the containing cone is adopted as the parent cone. It is therefore guaranteed that all quantities appearing in (4) lie within the valid range.

BVH generation

Since the number of BVH subtrees that need to be tested for self-collisions is determined by the NBCs, the same are used to influence how the BVH is generated and updated, so as to reduce the number of BV intersection tests and updates. The generation process comprises two main stages, in the first of which, disconnected geometry is identified and the top part of the BVH is created from the boxes bounding these clusters, bottom-up (Fig. 3). The second stage is the top-down division

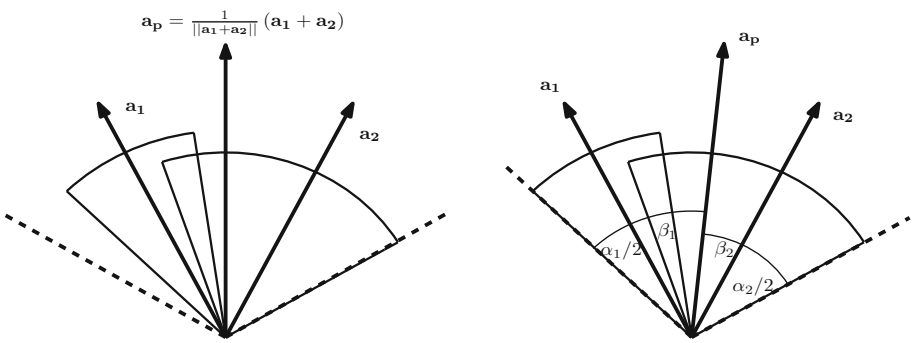


Fig. 2 Provot's NBC (*left*) and narrowest NBC (*right*). Child cones drawn with *solid lines*, parent NBC with *dashed lines*

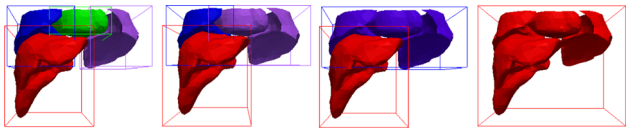


Fig. 3 Illustration of the bottom-up BV merging process. The *leftmost picture* shows the initial state when the AABBs contain only connected geometry. The *next picture* shows the state after the *two boxes* yielding the smallest parent box have been merged. The *last picture* (rightmost) shows the BVH root bounding all geometry. Meshes courtesy of IRCAD (http://www.ircad.fr/softwares/3Dircadb/3Dircadb.php?lng=en)

of the boxes bounding the connected primitives. In order to be able to apply the NBC self-collision criterion, the division process makes sure that the primitives contained in the newly created BVs remain connected. The cost function governing the assignment of primitives to child BVs, Eq. (5), consists of two quantities to be minimised: the volume of the resulting BV and the opening angle of the NBC.

$$\mathcal{B}^{n+1}_{\text{child}_i} = \mathcal{B}^{n}_{\text{child}_i} \cup \left\{ \arg \min_{T \in \mathcal{B}_{\text{parent}}} V(\mathcal{B}^{n}_{\text{child}_i} \cup \{T\})(1 + c \cdot \alpha_{\text{VMT}}(\mathcal{B}^{n}_{\text{child}_i} \cup \{T\})) \right\} \tag{5}$$

$\mathcal{B}_{\text{child}_i}$, $\mathcal{B}_{\text{parent}}$ denote the primitive sets bounded by the new children and the parent BV being split, respectively. $T \in \mathcal{B}_{\text{parent}}$ is any unassigned primitive from the parent-BV set, $V(\mathcal{B})$ is the volume of the BV bounding the primitive set $\mathcal{B}$, $\alpha_{\text{VMT}}(\mathcal{B})$ the opening angle of its NBC. To be able to mix volumes and angles, we introduce the constant c for which we determined $2/\pi$ to be a good value. The child-primitive sets $\mathcal{B}_{\text{child}_i}$ are initialised with the two primitives in the parent whose centroids are the farthest apart.

BVH updating

The BVH updating algorithm only refits bounding volumes to accommodate the deformation undergone by the geometry during a time step; it does not make any changes to the BVH topology. This selective updating is achieved by means of *update nodes* (UN) carried over from our previous work [18]. The UNs are defined as subtree roots in which the deformation undergone by the bounded geometry is quantified to assess the need for an update of the respective BVH subtree before the next collision detection pass. An update of the subtree is required, if 1) there are potential self-collisions in the subtree, or 2) the geometry has moved so much that the bounds of the subtree's BVs are no longer valid. The top part of the BVH, i.e. the part above the UN, is updated in every time step.

In order to evaluate criterion 1), a bound on the *non-rigid* deformation the geometry can undergo without causing an expansion of the NBC opening angle α_{VMT} beyond the threshold τ_{VMT} is required. The bound is given in (6) and derived in the "Appendix C".

$$\tau_{\text{NR}} = \frac{h_{\min}}{2} \tan \frac{\tau_{\text{VMT}} - \alpha^{(t_U)}_{\text{VMT}}}{2} \tag{6}$$

Where $h_{\min}$ is the minimum primitive diameter in the subtree. Finding it can be done recursively as part of the subtree update and at virtually no cost. The computation of the actual non-rigid displacement magnitude $||\boldsymbol{u}_{\text{NR}}||$ requires Procrustes analysis and is significantly more costly. However, the information obtained in the Procrustes analysis can be used to very cheaply update the subtree if update criterion 2) is satisfied but not 1), and the BV type supports such rigid body transforms, as do, e.g. oriented bounding boxes [9]. This gives rise to the update algorithm, Algorithm 1. The term ϵ_r, appearing in the pseudo-code, will be explained in section "Contact-force calculations".

Algorithm 1 BVH update algorithm

for all $\text{bv}_{\text{sub}} \in$ {Update nodes} **do**
  if $\alpha_{\text{VMT}}(\text{bv}_{\text{sub}}) \geq \tau_{\text{VMT}}$) **then**
    {Subtree has potential self-collisions}
    `RefitTopDown`(bv_{sub})
    `RefitBottomUp`(bv_{sub})
    $\boldsymbol{x_i}^{(t_U)} \leftarrow \boldsymbol{x_i}^{(t)}$ {Update reference nodes}
  else
    $\boldsymbol{c}^{(t)} \leftarrow \frac{1}{|\mathcal{N}(\text{bv}_{\text{sub}})|} \sum_{i \in \mathcal{N}(\text{bv}_{\text{sub}})} \boldsymbol{x_i}^{(t)}$ {Compute new centroid of bounded mesh nodes}
    $\boldsymbol{t} \leftarrow \boldsymbol{c}^{(t)} - \boldsymbol{c}^{(t_U)}$ {Compute subtree translation}
    $\max ||\boldsymbol{u}_{\text{NT}}|| \leftarrow \max_{i \in \mathcal{N}(\text{bv}_{\text{sub}})} ||\boldsymbol{x_i}^{(t)} - \boldsymbol{x_i}^{(t_U)} - \boldsymbol{t}||$ {Compute maximum non-translational motion}
    if $\max ||\boldsymbol{u}_{\text{NT}}|| < \tau_{\text{NR}}$ **and** $\max ||\boldsymbol{u}_{\text{NT}}|| + ||\boldsymbol{t}|| < \epsilon_{BV} - 2\epsilon_r$ **then**
      {Subtree inactive; motion insignificant, no update}
      `RefitBottomUp`(bv_{sub})
      continue
    end if
    if $\max ||\boldsymbol{u}_{\text{NT}}|| < \tau_{\text{NR}}$ **and** $\max ||\boldsymbol{u}_{\text{NT}}|| < \epsilon_{BV} - 2\epsilon_r$ **then**
      {Translational motion dominant}
      `ApplyTranslationTopDown`(bv_{sub}, $\boldsymbol{t}$)
      $\boldsymbol{x_i}^{(t_U)} \leftarrow \boldsymbol{x_i}^{(t_U)} + \boldsymbol{t}$ {Translate reference nodes}
      `RefitBottomUp`(bv_{sub})
      continue
    end if
    $R \leftarrow$ `Procrustes`$(\boldsymbol{x_i}^{(t_U)} - \boldsymbol{c}^{(t_U)}, \boldsymbol{x_i}^{(t)} - \boldsymbol{c}^{(t)})$ {Compute subtree rotation with Procrustes analysis}
    $\max ||\boldsymbol{u}_{\text{NR}}|| \leftarrow \max_{i \in \mathcal{N}(\text{bv}_{\text{sub}})} ||\boldsymbol{x_i}^{(t)} - \boldsymbol{c}^{(t)} - R(\boldsymbol{x_i}^{(t_U)} - \boldsymbol{c}^{(t_U)})||$ {Compute max. non-rigid deformation}
    if `DoesSupportRotation`(bv_{sub}) **and** $\max ||\boldsymbol{u}_{\text{NR}}|| < \tau_{\text{NR}}$ **and** $\max ||\boldsymbol{u}_{\text{NR}}|| < \epsilon_{BV} - 2\epsilon_r$ **then**
      {Rigid motion}
      `ApplyTransformTopDown`(bv_{sub}, R, $\boldsymbol{t}$)
      $\boldsymbol{x_i}^{(t_U)} \leftarrow R(\boldsymbol{x_i}^{(t_U)} - \boldsymbol{c}^{(t_U)}) + \boldsymbol{c}^{(t)}$ {Transform reference nodes}
    else
      `RefitTopDown`(bv_{sub})
      $\boldsymbol{x_i}^{(t_U)} \leftarrow \boldsymbol{x_i}^{(t)}$ {Update reference nodes}
    end if
    `RefitBottomUp`(bv_{sub})
  end if
end for

The UN role is assigned to BVs at the time of BVH creation. Since subtrees potentially containing self-collisions

always have to be updated, it makes sense to place the update nodes well below a point in the tree where $\alpha_{\mathrm{VMT}} > \tau_{\mathrm{VMT}}$. The algorithm for the placing is a greedy one, initialising the set of UNs with the leafs of the BVH. In every iteration, it picks the two nodes whose parent has the narrowest NBC opening angle and replaces them with their parent in the intermediate set of update nodes. This procedure continues until the minimum value of α_{VMT} of the set of parent BVs of the current UN exceeds a threshold, set to $\frac{1}{2}\tau_{\mathrm{VMT}}$ in the implementation.

Collision detection

The broad-phase collision detection is performed by recursively checking BVH (sub-) trees against each other until the bottom of the two BV trees is reached, i.e. the BVs bounding individual geometrical primitives [25]. For self-collision detection, the children of any BVH node where Eq. (3) holds true needs to be checked against each other. The subsequent primitive–primitive test consists of one test for vertices against facets and one for edges against edges. The node-facet test starts with the computation of an initial projection onto the master surface and gap value, $\left(\tilde{\xi}, \tilde{\eta}, \tilde{g}\right)$, for the predicted position of the slave vertex $\boldsymbol{x_s}$. The initial projection is obtained with Möller and Trumbore's method [27] with the minor modification of employing normalised facet normals instead of unnormalised one. This particular method is only applicable with triangular surface discretisations.

If the initial-guess projection $(\tilde{\xi}, \tilde{\eta})$ lies within the bounds of the master facet and the initial guess for the gap value, $|\tilde{g}|$, is sufficiently close to the previously nearest projection, it is improved upon with an iterative procedure that employs *C0*-continuous master facet normals $\boldsymbol{n_m}(\xi, \eta)$, computed from the vertex normals obtained by averaging the normals of the incident facets. This yields the final penetration depth (gap function value), g, and projection $\boldsymbol{x_m}(\xi, \eta)$:

$$\begin{aligned} \boldsymbol{x_m}(\xi, \eta) &:= \sum_{i \in \{\text{master facet vertices}\}} b_i(\xi, \eta)\boldsymbol{x_i} \\ \Rightarrow g &:= \boldsymbol{n_m}^{\mathrm{T}}(\xi, \eta)(\boldsymbol{x_s} - \boldsymbol{x_m}(\xi, \eta)) \end{aligned} \tag{7}$$

where $\boldsymbol{x_i}$ denotes the coordinates of the i-th master facet vertex, and $b_i(\xi, \eta)$ is the corresponding standard 2D linear shape function.

This projection $\boldsymbol{x_m}(\xi, \eta)$ is the virtual master surface node based on which the contact forces are computed (section "Contact-force calculations"). For each slave node, only the nearest projection onto a master facet is stored.

The edge–edge collision detection is performed by determining for the potentially colliding edge–edge pairs turned up by the broad-phase search the points on the two edges with the smallest distance at the end of the time step. This yields the parameters q and $r \in [0, 1]$ that represent the position of the closest point on the slave and the master edge, respectively. The difference between those closest points is subsequently projected onto the master surface normal, resulting in the gap function for the edge–edge case:

$$g = \boldsymbol{n_m}^{\mathrm{T}}(r)(\boldsymbol{x_s}^{(t)}(q) - \boldsymbol{x_m}^{(t)}(r)). \tag{8}$$

Contact-force calculations

We distinguish two types of contact forces, penetration responses and gap rate proportional forces. The former come into effect if a predictor displacement configuration leads to intersections of master and slave surfaces. The latter slow down approaching master and slave surfaces before they can intersect. The derivation of the force formulations started from the work of Heinstein et al. [14], modifications on the side of penetration responses include the distribution of forces over the master surface, the extension to the edge–edge collision case, the momentum-preserving gap-partitioning factor formulation, and the force consolidation algorithm. The gap rate proportional forces that in practice constitute the bulk of the contact forces are a new formulation. Another major objective of the presented formulation is to, as far as possible, enforce contact constraints one-by-one and avoid the introduction of any matrix formulations in the force computation step, so as to keep the required communication between workers processing individual contacts at a minimum.

Penetration response calculation

Penetrations are mathematically characterised by $g < 0$, i.e. situations where the slave node lies behind the master surface facet with respect to the master surface-normal direction. The penetration response force formulas arise directly from Eq. (1) with the requirement of immediate resolution of any mesh intersection and, for *node-facet* penetrations, are given by

$$\begin{aligned} \boldsymbol{f_s} &= -\boldsymbol{n}(\xi, \eta)\beta_s \frac{m_s g}{\Delta t^2} \\ (\boldsymbol{f_m})_i &= \boldsymbol{n}(\xi, \eta)\beta_m \frac{m_i g \gamma_i(\xi, \eta)}{\Delta t^2}, \quad i \in \text{master facet} \end{aligned} \tag{9}$$

$\boldsymbol{f_s}$ is the force applied to the penetrating slave node, $(\boldsymbol{f}_m)_i$ denotes the force applied to one of the three vertices of the penetrated master surface primitive. m_s, m_i are the masses of the slave node and master facet nodes, respectively.

The factor γ_i

$$\gamma_i(\xi, \eta) := \frac{b_i(\xi, \eta)}{\sum_{j \in \{\text{master facet vertices}\}} b_j(\xi, \eta)^2}, \quad i \in \text{master facet} \tag{10}$$

distributes the gap function value over the master facet such that at the position $\boldsymbol{x_m}$, the fraction of the gap assigned to the master surface is recovered [23]:

$$\beta_m g = \sum_{i\in\{\text{master facet vertices}\}} b_i(\xi,\eta)\gamma_i(\xi,\eta)\beta_m g \tag{11}$$

The gap-partitioning factor β, appearing in (9), controls how the response is split between master and slave surfaces. For contacts between rigids and deformables, it is set to 0 and 1, respectively. For inter-penetration of deformables, it holds a value in]0, 1[that is computed from the masses of the nodes involved in the contact:

$$\beta_s = \frac{m_m}{m_s + m_m}, \quad \beta_m = 1 - \beta_s = \frac{m_s}{m_s + m_m} \tag{12}$$

For the purpose of gap partitioning, the mass of the virtual master node m_m is computed with linear interpolation from the corresponding facet-vertex masses.

With these definitions, it holds at the point of contact on the master surface:

$$\begin{aligned} \boldsymbol{f_m} &= \sum_{i\in\{\text{master facet vertices}\}} b_i(\xi,\eta)(\boldsymbol{f_m})_i \\ &= -\boldsymbol{f_s} = -\lambda\,\boldsymbol{n} \end{aligned} \tag{13}$$

where λ is the Lagrange-multiplier for the constraint.

Edge–edge penetration responses employ the same rationale that underlies Eq. (9), except that there are now two slave nodes and only two master nodes, and the 2D shape functions of (9) are now 1D ones.

$$\begin{aligned} (\boldsymbol{f_s})_i &= -\boldsymbol{n_m}(r)\beta_s \frac{m_i\gamma_i(q)g}{\Delta t^2}, \; i \in \text{slave edge} \\ (\boldsymbol{f_m})_i &= \boldsymbol{n_m}(r)\beta_m \frac{m_i\gamma_i(r)g}{\Delta t^2}, \; i \in \text{master edge} \end{aligned} \tag{14}$$

Gap rate proportional forces

The gap rate proportional forces are employed to achieve a degree of temporal smoothing in the contact forces and thus to improve stability. They come into effect when $0 \le g < \epsilon_r$ and the relative velocity between slave and master, in master normal direction, the *gap rate*, is negative:

$$\begin{aligned} &\boldsymbol{n_m}^{\mathrm{T}}(\boldsymbol{v_s} - \boldsymbol{v_m}(\xi,\eta)) < 0 \\ &\boldsymbol{v_s} := (\boldsymbol{x_s}^{(t)} - \boldsymbol{x_s}^{(t-\Delta t)})/\Delta t, \\ &\boldsymbol{v_m}(\xi,\eta) := (\boldsymbol{x_m}^{(t)}(\xi,\eta) - \boldsymbol{x_m}^{(t-\Delta t)}(\xi,\eta))/\Delta t \end{aligned} \tag{15}$$

The constant $\epsilon_r = \frac{5}{100\cdot 2}\epsilon_{BV}$ is chosen such that any node at distance ϵ_r from a master facet still lies within the safety margin of the BV and so close that any effects of the force applications are not visible in the final configuration.

The force required for velocity-matching of the slave node and the virtual master node is derived from the forward-Euler increment of the velocity and momentum conservation as follows:

$$\begin{aligned} &\boldsymbol{n_m}^{\mathrm{T}}[(\boldsymbol{v_s} - \boldsymbol{v_m}) + (\boldsymbol{\Delta} v_s - \boldsymbol{\Delta} v_m)] \stackrel{!}{=} 0 \\ &\boldsymbol{\Delta} v_s = \frac{\Delta t}{m_s}\boldsymbol{n_m}\dot{\lambda}, \\ &\boldsymbol{\Delta} v_m = -\sum_{i\in\text{master facet}} b_i \frac{\Delta t}{m_i}\gamma_i \boldsymbol{n_m}\dot{\lambda} \end{aligned} \tag{16}$$

This gives rise to the following formula for the force's magnitude $\dot{\lambda}$

$$\dot{\lambda} = -\frac{\boldsymbol{n_m}^{\mathrm{T}}(\boldsymbol{v_s} - \boldsymbol{v_m})}{\Delta t(1/m_s + \sum_i b_i\gamma_i/m_i)} \tag{17}$$

The applied force gradually increases as the distance between the surfaces decreases, and full velocity-matching is performed when there is zero distance between the slave node and its projection onto the master surface

$$\begin{aligned} \boldsymbol{f_s} &= (1 - g/\epsilon_r)\boldsymbol{n}(\xi,\eta)\dot{\lambda}, \\ (\boldsymbol{f}_m)_i &= -(1 - g/\epsilon_r)\boldsymbol{n}(\xi,\eta)\dot{\lambda}\gamma_i \end{aligned} \tag{18}$$

The edge–edge contact formula for $\dot{\lambda}$ reads:

$$\dot{\lambda} = -\frac{\boldsymbol{n_m}^{\mathrm{T}}(\boldsymbol{v_s} - \boldsymbol{v_m})}{\Delta t\left(\sum_{i\in\text{master edge}} b_i\gamma_i/m_i + \sum_{i\in\text{slave edge}} b_i\gamma_i/m_i\right)} \tag{19}$$

Friction

Equation (17) can be used in Coulomb's model to simulate friction by substituting the relative tangential velocity for the gap rate. Modelling friction only requires keeping track of the active constraints in a given time step and the associated normal forces, and application of the forces computed from

$$\boldsymbol{f} = \begin{cases} -\lambda_T\,\boldsymbol{\Delta} v_T/||\boldsymbol{\Delta} v_T||, & \text{if } \lambda_T < \mu||\boldsymbol{f_N}|| \\ \mu||f_N||\boldsymbol{\Delta v_T}/||\boldsymbol{\Delta} v_T||, & \text{otherwise} \end{cases} \tag{20}$$

with μ being the friction coefficient, $\boldsymbol{f_N}$ the normal forces applied to the corresponding slave node, and

$$\begin{aligned} \boldsymbol{\Delta} v_T &:= (\boldsymbol{v_s} - \boldsymbol{v_m}) - \boldsymbol{n_m}^{\mathrm{T}}(\boldsymbol{v_s} - \boldsymbol{v_m})\cdot\boldsymbol{n_m} \\ \lambda_T &:= \frac{||\boldsymbol{\Delta} v_T||}{\Delta t(1/m_s + b_i\gamma_i/m_i)} \end{aligned} \tag{21}$$

Contact-force consolidation

Since nodes can be involved in multiple contacts, e.g. multiple edge–edge contacts or master facet vertices being part of multiple node-facet contacts, the contact forces must be consolidated. This can be relatively easily accomplished if the indices subject contact constraints are being kept track of as the responses are computed. The option chosen in our algorithm is that of computing for every node with an active contact constraint the mean direction of the contact forces and applying the maximum projection over all response forces in that direction. Algorithm 2 contains a pseudo-code description of the consolidation algorithm.

Fig. 4 Simulation geometry and settings used in validation of the proposed cone formula

Zipper: "Zipper" geometry twisted and compressed through force boundary conditions ($F_x = \pm 30,\ F_y = \pm 5,\ F_z = -10$). Initial configuration shown in left, final configuration shown in right picture.
Dimensions: $35 \times 7 \times 12.5$.
Mesh: 712 surf. facets, 358 surf. nodes, 1587 solid el's, 485 solid nodes.
Material: $G = 10$, $K = 40$, $\rho = 100$.
Total time: $T = 100$.
Time step size: $\Delta t = 0.075$.

C: C shape being closed by application of displacement boundary conditions to upper part of geometry ($u_z = -3.5$).
Dimensions: $22.5 \times 10 \times 21.5$.
Mesh: 960 surf. facets, 482 surf. nodes, 2320 solid el's, 667 solid nodes.
Material: $G = 10$, $K = 40$, $\rho = 1$.
Total time: $T = 2$.
Time step size: $\Delta t = 0.001$.

Algorithm 2 Algorithm for handling of redundant constraints.

for all $n \in$ {nodes with contacts} **do**
  $\overline{f} = \sum_{f \in \{\text{contact forces applied to } n\}} f / || \sum_{f \in \{\text{contact forces applied to } n\}} f ||$
  {Compute (weighted) mean direction of contact forces}
  $f_n \leftarrow \max_{f \in \{\text{contact forces applied to } n\}} f^{\mathrm{T}} \overline{f}$ {Compute maximum projection along mean direction}
  $f_n \leftarrow f_n \overline{f}$
end for

With the right data structures, this algorithm can be implemented with an $O(N_{\text{node-facet}} + N_{\text{edge-edge}})$ runtime complexity, where $N_{\text{node-facet}}$ and $N_{\text{edge-edge}}$ are the number of node–facet and edge–edge contacts, respectively.

Experiments

Our objective in this section was to show the inherent advantages of the individual heuristics introduced in this paper over a number of alternatives. The subsequent evaluations are based on a single-threaded C++ implementation of the algorithm described in the previous section. The FEM calculations were done with NiftySim's CPU solver [19]. The timings were obtained on a workstation equipped with an Intel Core i7 2,600K processor and 8GB of RAM, by surrounding individual parts of the code representing the major stages of the contact-modelling pipeline with calls to the C `clock` function and accumulation.

Self-collision detection cones

The first two experiments are aimed at quantifying the effects of the formula for computation of the NBCs described in section "Self-collisions and surface-normal bounding cones". To this end, it is compared with the method of Provot [29]. The geometry and simulation settings of the simulations are given in Fig. 4. Due to the artificial nature of the geometry, units have been omitted, but all parameter values can be assumed to be specified in compatible units, e.g. m, s, Pa. The geometry was generated with AutoCAD[2], and solid meshing was done with GMSH.[3] Most of the experiments are once run with a binary AABB hierarchy (AABB2) and once with a 4-nary BVH (AABB4).

The results in terms of the average number of BVH subtrees that need checking for self-collisions per time step, BV refittings, and the total time spent updating BVHs and searching for contacts, for these two simulations are given in Table 1.

A reduction in the 40 % ballpark in the number of BVH subtrees that needs to be visited for self-collision detection (first column in Table 1) is achieved with our proposed formula in these two test cases. Further, since our BVH refitting and construction algorithms take into account the NBC opening angle, the number of BVs that are refitted is about 10 % lower with our NBC computation method (second column in Table 1). The latter effect can mostly make up for the slight increase in computational costs that comes with our formulation. These findings are consistent across the two considered BVH orders.

BVH refitting strategy

The second set of experiments deals with the evaluation of the proposed BVH updating strategy. Most of the geometry used in these experiments is again artificial and created with Meshlab,[4] except the test case containing a liver and diaphragm whose geometry was extracted from volunteer MRI data with Slicer 3D.[5] The "C" test case from the previ-

[2] http://usa.autodesk.com/autocad/.
[3] http://www.geuz.org/gmsh/.
[4] http://meshlab.sourceforge.net/.
[5] http://www.slicer.org/.

Table 1 Results from the comparison of our cone computation method with that of Provot

Experiment	Avg. No. of BVH subtrees with potential self-collisions	Avg. No. of refitted BVs/time step (tot. No. of BVs)	Contact search total time (s)	BVH update total time (s)
(a) Our cone method				
C AABB2	32.9	216.2 (1,919)	4.35	0.129
C AABB4	17.4	101.4 (1,410)	5.01	0.117
Zipper AABB2	50.3	229.4 (1,423)	1.43	0.0936
Zipper AABB4	22.9	167.3 (1,065)	1.49	0.1
(b) Provot's method				
C AABB2	57.5	234.5 (1,919)	5.04	0.113
C AABB4	28.5	121.3 (1,410)	5.49	0.103
Zipper AABB2	87.7	254 (1,423)	1.62	0.0823
Zipper AABB4	45.2	220.7 (1,065)	1.72	0.0898

ous section is also used in these experiments. The new test simulations are summarised in Fig. 5.

The results for these experiments can be found in Table 2. No results are available for Larsson and Akenine-Möller's method on the "rigid bar" test case, since the method is not defined for rigid-deformable contacts. Similarly, while technically it can be easily extended to applications in self-collision detection, its inventors never intended for it to be used in that way, and its performance is very poor and provides little insight in this context. Therefore, there are no results for the "C" test case in Table 2b, either.

The first observation that can be made from these results is that our proposed BVH updating strategy leads to a general reduction in BV refitting and, ultimately, in overall computation time, over both exhaustive refitting and Larsson and Akenine-Möller's update strategy. This effect is more pronounced on higher resolution meshes and problems not involving self-collisions. The latter is most likely due to the dominant effect of contact search on the overall computation costs of self-collision problems. On the larger problems, the reduction in BVH refitting costs over exhaustive refitting approaches one order of magnitude with our method, despite the not insignificant computational overhead introduced by the deformation analysis.

Scaling

The aim here was to show how the performance of our proposed NBC computation formula and BVH update method change with increasing mesh resolution. To this end, the zipper self-collision test case is taken and the mesh refined in 5 steps. The Zipper test case was chosen for this experiment due to the relatively large deformation, and because all deformation stems from force application, there is hence no bias towards translational movement of geometry, which might give our proposed method an unfair advantage.

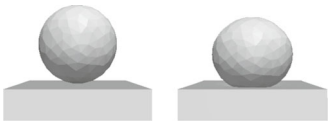
The first experiment looks at the NBCs. The same simulation is run once with Provot's method and once with ours and the average number of BVH subtrees that need checking for self-collisions and the average time required for contact-modelling operations per time step are recorded. The results for binary and 4-nary AABB hierarchies can be found in Fig. 6. The timing values include the time required for all BVH and contact surface update, as well as collision detection and response computations.

There is a clear divergence in the number of sub-trees that need checking for self-collisions between the two methods, with noticeably lower computation times for our method at higher mesh resolutions. That the divergence in the time required for contact modelling operations, in turn dominated by the contact search, does not diverge stronger can likely be explained with the subtree pairs that need checking with both types of NBCs having their roots higher up in the hierarchy and thus being more time-consuming to traverse.

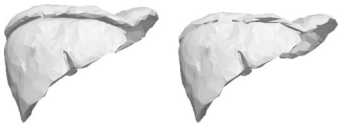
The second experiment, looking at the proposed BVH update strategy, is mostly identical except what is recorded is the number of refitted BVs and the average per time-step BVH refitting costs in milliseconds. The results for exhaustive refitting are included for reference. The results for binary and 4-nary AABB hierarchies can be found in Fig. 7.

The most striking result is, as the number of surface primitives increases almost ten-fold, the number of updated BVs remains almost constant with our strategy. This can be explained with the UN being placed based purely on geometric criteria. Meanwhile, the costs of exhaustively refitting the BVH approaches 50 % of the total contact-modelling costs observed for our algorithm in Fig. 6. From these plots, it can also be seen that the savings in terms of overall computation time with AABB4 over AABB2 and exhaustive refitting, observable in Table 2, can be attributed only to the fewer refitted BVs with the higher order BVH.

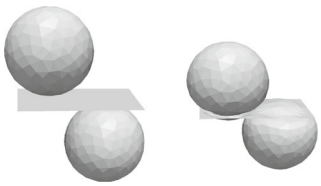
Fig. 5 Experiments used in BVH update-strategy validation



Ball-slab: Ball being pushed against a deformable slab through displacement boundary conditions (vertically by $u_z = -0.4$).
Dimensions: Ball radius = 1.2, slab: $3.15 \times 3.15 \times 1.2$.
Mesh: 1500 surf. facets, 754 surf. nodes, 3284 solid el's, 981 solid nodes.
Material: $G = 100$, $K = 400$, $\rho = 20$.
Time: $T = 2$, $\Delta t = 0.001$.



Diaphragm/liver: Diaphragm pushed against liver by displacement boundary conditions ($u_z = -0.2$).
Mesh: 2400 surf. facets, 1202 surf. nodes, 4921 solid el's, 1448 solid nodes.
Material: $G = 15$, $K = 50$, $\rho = 200$.
Time: $T = 1$, $\Delta t = 0.0025$



Two spheres: Two spheres pushed against a membrane via force constraints ($F_z = \pm 1$), leading to two-sided contact. Only membrane mid-surface shown.
Dimensions: sphere radius = 0.7, membrane dimensions = $2 \times 1 \times 0.15$.
Mesh: 2×700 sphere surf. facets, 2×252 sphere surf. nodes, 2×789 solid el's, 2×278 solid nodes, 400 membr. el's, 231 membr. nodes.
Material: $G_{\text{sphere}} = 200$, $K_{\text{sphere}} = 600$, $G_{\text{membrane}} = 1200$, $\rho_{\text{sphere}} = 25$, $\rho_{\text{membrane}} = 200$; membrane is incompressible.
Time: $T = 0.5$, $\Delta t = 10^{-4}$

Rigid bar: An irregularly shaped body (grey) is pushed via displacement boundary conditions ($u_x = 0.5$) on its back against a rigid bar (semi-transparent, blue).
Dimensions: $0.91 \times 1 \times 1.34$ (deformable body AABB), $0.5 \times 0.25 \times 0.25$ (rigid body)
Mesh: 600 surf. facets, 302 surf. nodes, 1184 solid el's, 359 solid nodes, 120 rigid facets, 83 rigid nodes.
Material: $G = 0.05$, $K = 0.1$, $\rho = 0.5$
Time: $T = 2$, $\Delta t = 0.0005$

Contact forces

The next two experiments look at the quality of the contact forces by considering two geometrically well defined benchmark problems. The first experiment simulates a Newton's cradle consisting of 4 elastic spheres (Fig. 8). All spheres are the same size ($r = 1$), identically discretised (1,004 nodes, 5,101 tetrahedra), and have the same material parameters ($G = 1{,}000$, $K = 4{,}000$, $\rho = 10$). The leftmost ball is given an initial velocity of $\boldsymbol{v} = (0.1, 0, 0)^{\mathrm{T}}$, no other boundary conditions are applied. The simulation comprises 100,000 time steps of $\Delta t = 10^{-4}$ each for a total time of $T = 10$. The second one simulates the breaking of a billiard rack. Again the system consists of 4 balls, one of which is given an initial velocity of $v_x = 0.15$ (Fig. 8), and identical material parameters are used for all 4 balls in the experiment: $G = K = 1$, $\rho = 5$. The total simulated time is $T = 20$ with 20,000 time steps. To assess the energy conservation the time integration had to be performed with NiftySim's explicit Newmark time-ODE solver and without damping, since central-difference integration, even without damping, proved to be too dissipative.

The ball-centre (average node positions) trajectories and the corresponding energy balance of the system are given in Figs. 9 and 10 for the Newton's cradle and the billiard rack experiment, respectively. The strain energies in the plots are the sum of the internal energies of all elements in the simulation and the kinetic energies are computed through the inner product defined by the lumped mass matrix:

$$E_{\text{kin}} = \frac{1}{2\Delta t^2}(U^{(t)} - U^{(t-1)})^{\mathrm{T}} M (U^{(t)} - U^{(t-1)}) \tag{22}$$

In the Newton's cradle experiment, the ball-centre trajectories are all straight lines parallel to the x axis. The standard deviations of the ball trajectories in the y and z directions satisfy $\sigma < 1.7 \cdot 10^{-4}$. The mean velocity of ball 4 at the end of the simulation is $\boldsymbol{v} = (0.0996, 2.3 \cdot 10^{-4}, 1.2 \cdot 10^{-4})$. The total energy at the end of the simulation is 99.7 % of the initial energy. In the billiard experiment, the trajectories of ball 1 and 2 form straight lines with standard deviations in the y and z directions satisfying $\sigma < 4.35 \cdot 10^{-4}$. For balls 3 and 4, symmetric trajectories with slopes ± 0.82 are found. The respective deviation of the trajectories from that line are $\sigma_{\text{ball3}} = 3.3 \cdot 10^{-3}$ and $\sigma_{\text{ball4}} = 3.3 \cdot 10^{-3}$. The sum of all energy in the system at the end of the simulation is equal to the initial kinetic energy of ball 1. Calculated using a point mass approximation, the kinetic energy of balls 3 and 4 at the end of the simulation amounts to 96 % of the initial kinetic

Table 2 Results of comparison of BVH update strategies

Experiment	Avg. number of refitted BVs/time step (tot. No. of BVs)	BVH update: avg. time (ms)/time step	Total computation time (s)
(a) Our update strategy			
Ball-slab AABB2	111.9 (2,999)	0.0433	2.91
Ball-slab AABB4	50.1 (2,265)	0.0407	4.02
C AABB2	216.2 (1,919)	0.0675	5.73
C AABB4	101.4 (1,410)	0.0606	6.34
Liver-diaphragm AABB2	659.7 (4,799)	0.202	1.65
Liver-diaphragm AABB4	412.4 (3,659)	0.196	1.67
Two spheres AABB2	160.5 (3,599)	0.0488	5.81
Rigid bar AABB2	75.3 (1,199)	0.0229	1.15
Rigid bar AABB4	44.7 (904)	0.0228	1.14
(b) Larsson and Akenine-Möller's update strategy			
Ball-slab AABB2	898.9 (2,999)	0.129	3.11
Ball-slab AABB4	581.9 (2,265)	0.0973	4.14
Liver-diaphragm AABB2	2,729.6 (4,799)	0.430	1.77
Liver-diaphragm AABB4	2,035.7 (3,659)	0.348	1.67
Two spheres AABB2	1,039.1 (3,599)	0.143	6.31
(c) Exhaustive refitting			
Ball-slab AABB2	2,999 (2,999)	0.301	3.49
Ball-slab AABB4	2,265 (2,265)	0.257	4.84
C AABB2	1,919 (1,919)	0.214	6.17
C AABB4	1,410 (1,410)	0.184	6.66
Liver-diaphragm AABB2	4,799 (4,799)	0.657	1.87
Liver-diaphragm AABB4	3,659 (3,659)	0.542	1.76
Two spheres AABB2	3,599 (3,599)	0.408	7.61
Rigid bar AABB2	1,199 (1,199)	0.146	1.65
Rigid bar AABB4	904 (904)	0.125	1.56

energy of ball 1, and 1.7 % are stored as strain energy in the balls.

In both experiments, the momentum and kinetic energy are almost fully transferred to the last balls in the chain. The method's ability to conserve momentum can be easily discerned in the Newton's cradle experiment, where the initial velocity of ball 1 is recovered in ball 4, at the end of the experiment. In the second experiment, the expected symmetric, linear trajectories for ball 3 and 4 are obtained. There is a small kink at the start of both trajectories most likely caused by a short phase in the experiment during which strain energy stored in ball 2 is converted back into kinetic energy and transferred to the last two balls and during which the three balls remain in continuous contact. There is a minor increase in the total energy during phases of conversion of kinetic energy into deformation. We experimentally excluded the temporal smoothing heuristic and the time discretisation as the cause, but did notice a correlation with the mesh density; a reduction in the number of elements per ball from 5,101 to 1,289 led to an increase of the highest peak from 105 % the initial energy to 109 %, in the billiard experiment. In any case, since this excess energy is absorbed just as smoothly as it arises and its magnitude in all experiments is in the single digits of per cents, it can be assumed to be harmless. Especially considering that the experiments had to be run with explicit Newmark time integration which the contact model was not explicitly designed for, these results are very satisfactory.

Friction

The following experiment is taken from Ref. [15]. It is a quasi-static friction problem with an analytical solution consisting of a bar being pushed down against a rigid surface, then dragged along the same via an outward pressure applied to its front face. The geometry of the experiment and location of boundary conditions is depicted in Fig. 11. The bar geometry is discretised with an unstructured mesh consisting of 1,678 nodes and 8,419 tetrahedra and the floor consists of 400 triangles and 231 nodes in a regular grid. The material of the bar is characterised by a Young's modulus of $E = 68.947$ MPa and a Poisson ratio of $\nu = 0$. The displace-

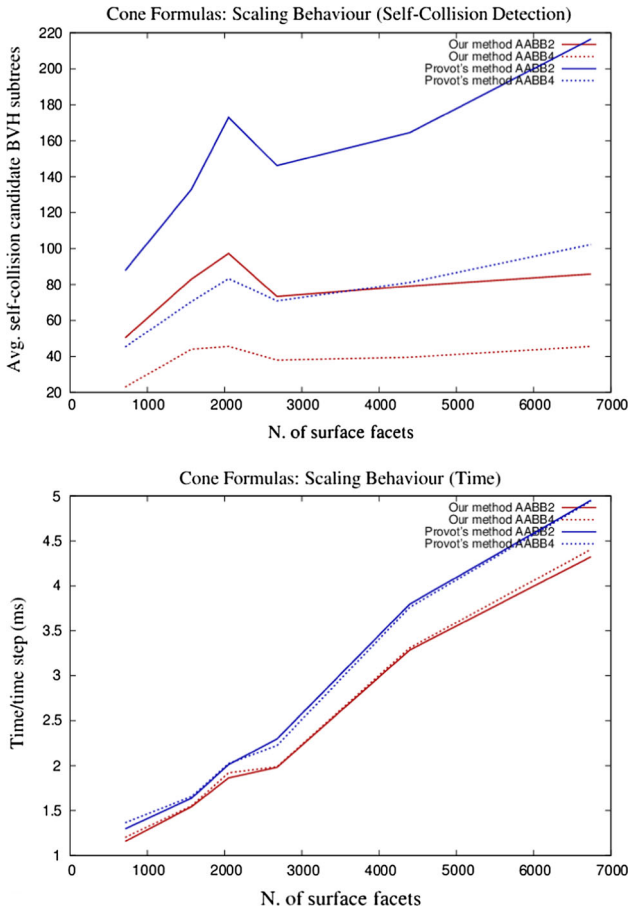


Fig. 6 Comparison of our method of NBC computation to that of Provot. *Top*: number of self-collision candidate subtrees plotted against mesh resolution. *Bottom*: corresponding contact modelling computation time per time step

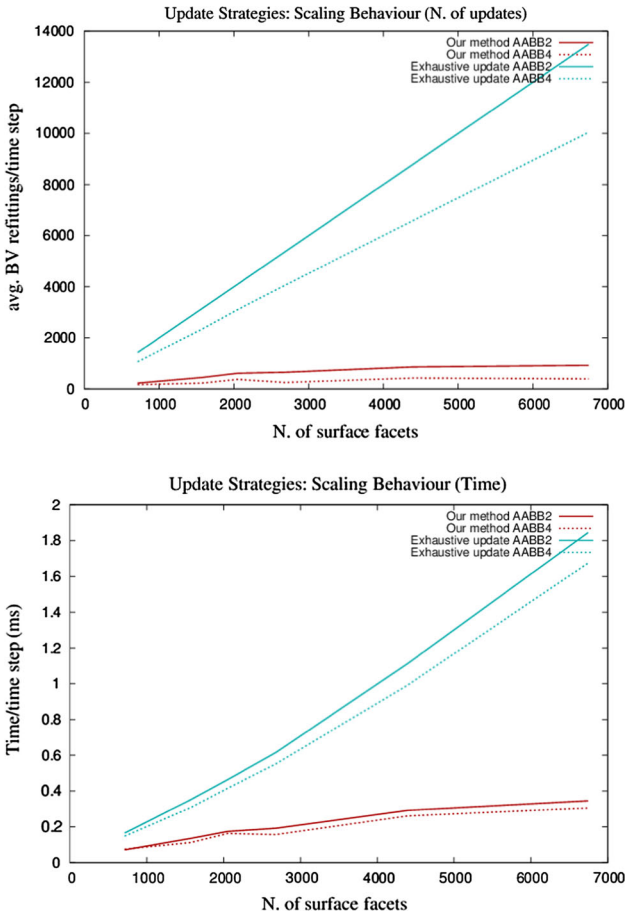


Fig. 7 Update-strategy scaling behaviour: our method, and exhaustive refitting. *Top*: average number of refitted BVs/time step. *Bottom*: average BVH update time/time step

ment boundary condition is ramped up during the first 1 % of the simulation time, the pressure is applied subsequently and linearly ramped up over the 99 % remaining time. The friction coefficient between the bar and the rigid surface is $\mu = 0.1$.

The final configuration is shown in Fig. 11. Figure 11 also shows a plot of the accumulated axial slip as a function of the corresponding point's x coordinate, at multiple time points in turn corresponding to different applied pressures. At most time points, the solutions for the tip displacement agrees with the analytic solution up to 5 %, a larger error of 8.2 % occurs at $t = 0.7$ after all but the constrained nodes have started slipping. At the end of the simulation, the measured peak displacement is $u_x = 1.287$ mm which is within 2 % of the analytical solution. If we set the displacement threshold to 0.005 mm, the same value of 317.5 mm is recovered for the slip/stick boundary at $t = 0.3$, as found by Heinstein and Laursen [15].

Given the relative simplicity of the friction model, the numerical result shows good agreement with the analytical solution. This result also provides evidence that the contact model is able to accurately deal with sustained contacts.

Examples from image-guidance

In this section, two quasi-static image-guidance examples of FEM contact modelling based on actual patient data are presented with the primary aim of demonstrating the proposed algorithm's performance on high-resolution meshes encountered in TLED's main application area. The code is sequential and uses binary AABB hierarchies for contact search. The first one is a reconstruction of the deformation caused to the prostate by the transrectal ultrasound (TRUS) probe used in guidance of needle biopsy and ablation procedures of prostate cancer. Being able to determine this deformation is crucial for the registration of the interventionally acquired TRUS images to MR images acquired prior to the procedure [17].

The anatomical meshes were generated from a $320 \times 320 \times 15$-voxel MR image with a $0.8 \times 0.8 \times 4$ mm resolu-

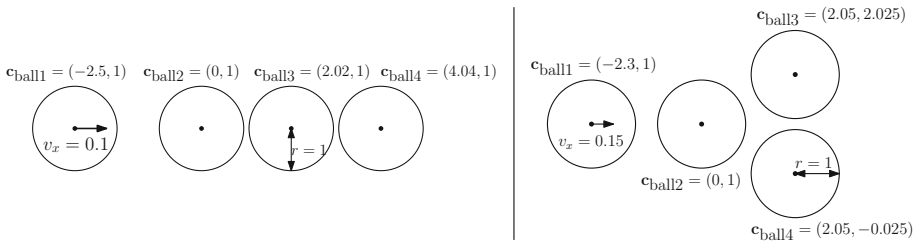


Fig. 8 *Left*: Newton's cradle with 4 balls; experiment initial configuration. *Right*: experimental setup: breaking of a 3-ball billiard rack

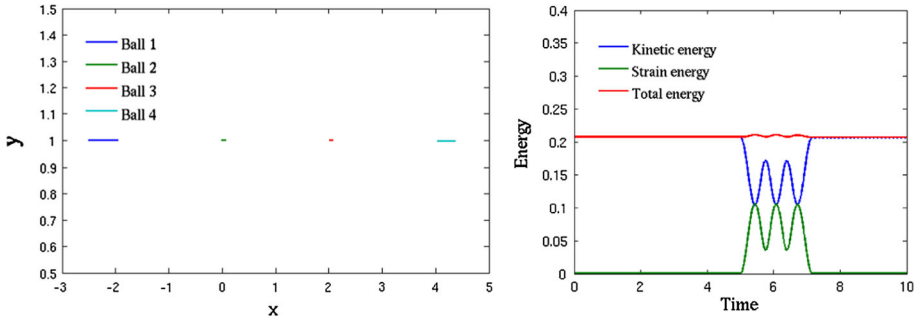


Fig. 9 *Left*: ball-centre x-y-plane trajectories for the Newton's cradle experiment. *Right*: plot of kinetic, strain, and total energy v. time for the Newton's cradle experiment

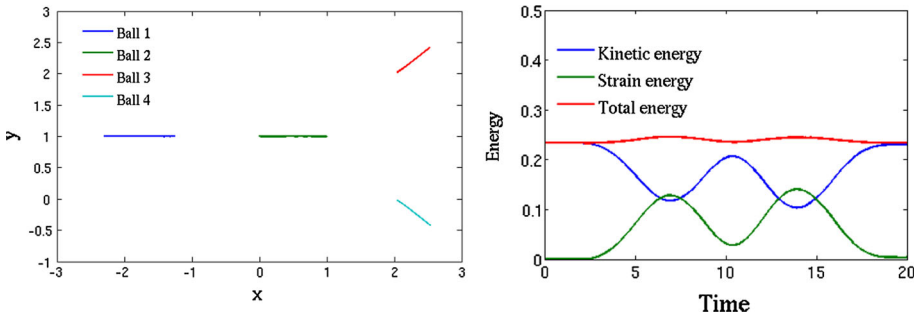


Fig. 10 *Left*: ball-centre x-y-plane trajectories for the billiard experiment. *Right*: plot of kinetic, strain, and total energy against time for the billiard experiment

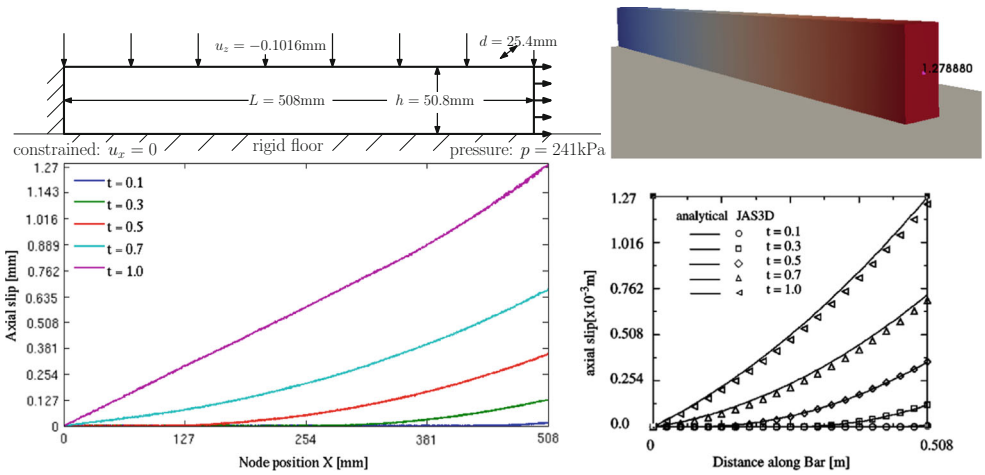


Fig. 11 *Top left*: experimental setup of the friction experiment. *Top right*: result configuration of friction experiment with u_x *colour mapped* and exact u_x value at centre of front face. *Bottom left*: plot of accumulated axial slip against nodal x coordinate. *Bottom right*: corresponding plot of axial slip found with JAS3D along with analytical solution, courtesy of Ref. [15]

tion with experimental, semi-automatic segmentation software and ANSYS.[6] The deformable geometry of the simulation consists of two unconnected parts: the prostate consisting of 22,705 tetrahedra and 4,425 nodes (purple in Fig. 12), and a block representing the surrounding tissue and the rectum consisting of 64,316 elements and 13,159 nodes (semi-transparent, blue in Fig. 12). The TRUS probe

[6] http://www.ansys.com/.

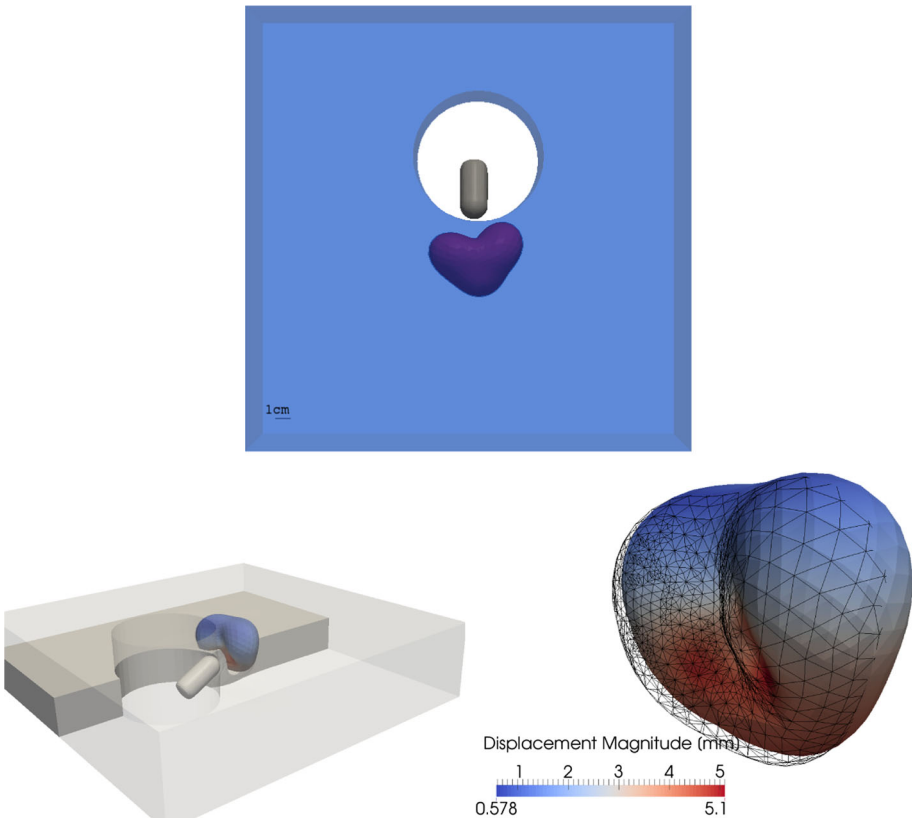


Fig. 12 *Top*: initial-configuration geometry of the prostate example. Prostate shown in *purple*, surrounding tissue in *blue*, TRUS probe mesh in *grey*. *Bottom left*: cutaway view of simulation final configuration. *Bottom right*: prostate final-configuration posterior 3/4 view with initial configuration overlaid (wireframe)

mesh was created in Meshlab and comprises 4,886 triangular elements and 2,445 nodes. Hu et al. [16] randomly sampled their material parameters from ranges given by $E \in [5, 150]$kPa, $\nu \in [0.3, 0.4999]$ for the prostate components and $E \in [5, 100]$kPa and $\nu \in [0.25, 0.499]$ for the surrounding tissue, and used an inhomogeneous model for the prostate and the surrounding tissue. The parameter values chosen for this purely demonstrational simulation are identical for the prostate and the other tissue and set to $G = 1.8$ kPa, $K = 6.9$ kPa. The front and back face of the block are fixed in all spatial directions via displacement boundary conditions. The probe is translated by $(-0, -11.5, 5)$ mm from its initial to final position, in a linear motion. The simulation runs for a total of 1,000 time steps of $\Delta t = 10^{-3}s$, each.

The second example is motivated by the registration of preoperatively acquired prone MR images used for planning of breast conserving cancer surgery to intra-operatively acquired supine MR images [12]. Sliding between the skin and underlying tissues has been observed but not properly quantified, having a method that allows for the modelling of this behaviour could therefore be used in future biomechanics-based registration algorithms for this type of application. In this example, the skin is modelled with a separate membrane mesh. The solid mesh comprises 37,613 tetrahedral elements and 10,614 nodes, and was generated from a $256 \times 512 \times 32$-voxel MR image with a $0.7 \times 0.7 \times 3.7$ mm resolution with experimental segmentation software and TetGen.[7] The membrane mesh was generated by extraction of the surface of the solid mesh, offsetting by 3 mm, and performing a manual segmentation; it has 4,425 elements, 2,283 nodes. The thickness of the shell elements is set to 5 mm leaving a 0.5 mm gap between the two meshes, that is quickly closed by the applied gravity forces. The chest-wall side of the solid mesh is fully fixed. The skin mesh is only held in the superior, lateral corner. The solid mesh is modelled as homogeneous transversely isotropic neo-Hookean [12] with $G = 3.57$ kPa, $K = 16.67$ kPa, $\eta = 37.71$kPa[8] and the preferential direction coinciding with the ventro-dorsal axis. The skin's material parameters are $E = 25$ kPa, $\nu = 0.4$ for the membrane component, $E = 5$ kPa, $\nu = 0.25$ for the bending stiffness. The simulation comprises 2,500 time steps, representing 2.5 seconds. For reference, an otherwise identical second simulation is run without the skin mesh to better quantify the effects of the skinning. A colour map of the distance between the two results can be seen in Fig. 13.

The deformation in the first experiment (Fig. 12) is, as can be expected, quite localised with the TRUS probe penetrating into the block by about 1.1 cm. In the process, the prostate is primarily rotated but also slightly bent with respect

[7] http://tetgen.berlios.de/.

[8] η controls the stiffness in the preferred material direction, details can be found in the NiftySim user manual.

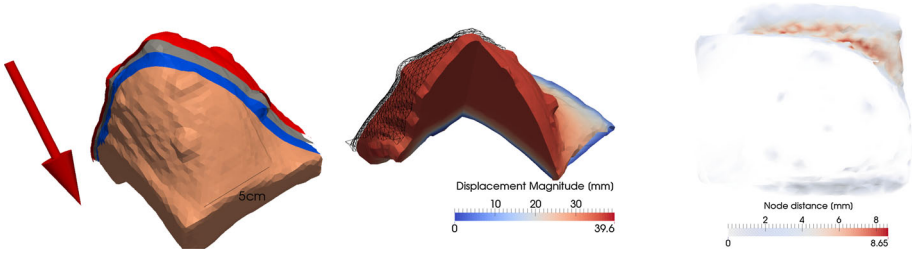


Fig. 13 *Left*: initial configuration of the breast simulation with the skin contact assembly partially peeled away for better visibility, showing the outer contact surface (*red*), midplane (*grey*, not used in contact modelling), and the bottom part of the membrane contact assembly (*blue*), and the breast solid mesh (beige). The *red arrow* indicates the direction of gravity. *Centre*: inferior-medial view of solid mesh and skin mid-surface final configurations with the skin mid-surface shown as wireframe mesh. *Right*: frontal view of skinned breast simulation final result with colour and opacity mapped distances to the result of the simulation without skin

to its apex-base axis with the base being displaced by about 4 mm. Two small dents made by the surrounding tissue displaced by the probe can also be seen on the prostate's posterior surface, where the peak displacement magnitude reaches 5.1 mm. That the non-rigid deformation of the prostate is not larger can probably be attributed to the mesh consisting of two parts that can slide relative to each other.

The deformation of the solid mesh in the breast example (Fig. 13) is primarily a compression in ventro-dorsal direction combined with a shift of a sizeable portion of the mass in inferior and medial direction. The skin mesh is well held in place by the former despite there only being displacement boundary conditions on one corner of the mesh. The mean solid mesh node distance between the results of the simulations with and without skin is 0.28 mm with a maximum of 8.61 mm. Most of the large magnitude interaction between the two meshes appears to happen in the area surrounding the breast, although there is some evidence of a constraining of the solid mesh's expansion in the plane of the chest wall. Further, due to the proximity of the skin and the solid mesh, roughly half of both the solid mesh and skin contact surface can be assumed to be subject to contact constraints, or at least be turned up as collision candidates by the broad-phase contact search, for most of the duration of the simulation which is a larger fraction than in most simulations. Thus, it can be assumed that particularly the contact search costs are higher in this simulation than in most simulations with a comparable mesh resolution.

In any case, the sum of the timings (Table 3) obtained for all contact modelling-related operations is in both cases of the same magnitude as the time required for the basic FEM modelling, which due to the low computational costs associated with the matrix-free approach of TLED is quite challenging in its own right.

Table 3 Computation times for the breast and prostate image-guidance examples broken down into the major stages of the contact-modelling pipeline

	Prostate (s)	Breast (s)
BVH and contact surface update	0.3	2.87
Contact search	45.55	119.71
Response computation	0.35	0.09
Other FEM operations	40.63	57.58
Total sim. computation time	86.83	181.06

Conclusions

We have presented methods suitable for detecting and handling of contacts arising in explicit FEM simulation of a range of scenarios: deformable geometry self-collisions, contacts between deformable solid and membrane meshes, and deformable geometry and a range of rigid geometry. The contact search portion of the presented pipeline is optimised for the typically small time steps one has with explicit time integration in which it keeps the number of BV refittings low by identifying the parts of the BVH where containment of the geometry is ensured and self-collisions can be excluded. The success of this strategy can be seen in the consistently low numbers of BV refittings.

Further, in this paper, an improved formula for the computation, via Provot's recursive algorithm, of surface-normal bounding cones used for self-collision detection has been presented. We have demonstrated that the proposed formula leads to a marked reduction in the number of BVH subtrees that must be checked for self-collisions, compared with the formula originally proposed by Provot.

On the contact-modelling side, we have presented a robust general-purpose method that can deal with both geometric as well as temporal singularities via smoothing. Mathematically, the contact-force smoothing is, with respect to space, done by means of linearly interpolated surface normals, and with respect to time, by means of linearly increasing gap rate proportional forces. Despite these stability improving modifications, the results obtained with the proposed contact model remain consistent with physics and accurate as has

been shown with transient impact and quasi-static, resting-contact friction experiments.

We have also shown that the entire proposed contact-modelling pipeline can be executed within a time frame that is of the same order of magnitude as the time required for standard TLED computations, in real-world image-guidance applications.

Acknowledgments This work was partially funded by the PASSPORT Liver (EU FP7) grant, and in part by the Intelligent Imaging Programme Grant (EPSRC Reference: EP/H046410/1). Z.A. Taylor is funded by the EU-FP7 project VPH-DARE@IT (FP7-ICT-2011-9-601055). Sebastien Ourselin receives funding from the EPSRC (EP/H046410/1, EP/J020990/1, EP/K005278), the MRC (MR/J01107 X/1), the EU-FP7 project VPH-DARE@IT (FP7-ICT-2011-9-601055), the NIHR Biomedical Research Unit (Dementia) at UCL and the National Institute for Health Research University College London Hospitals Biomedical Research Centre (NIHR BRC UCLH/UCL High Impact Initiative).

Conflict of interest S. F. Johnsen, Z. A. Taylor, L. Han, Y. Hu, M. J. Clarkson, D. J. Hawkes, and S. Ourselin declare that they have no conflict of interest.

Appendix A: The time-step algorithm

$t \leftarrow t + \Delta t$ {Beginning of time-step}
$R \leftarrow \texttt{UpdateExternalForces}\,(t)$
$F \leftarrow \texttt{UpdateInternalForces}\left(U^{(t)}\right)$
$U^{(p)} \leftarrow \texttt{UpdateDisplacements}(U^{(t)}, F, R)$ {Compute displacement prediction with Eq. (1)}

$C_{\text{def}}^{(t)} \leftarrow \texttt{Update}\left(C_{\text{def}}^{(t-\Delta t)}, U^{(p)}\right)$ {Update deformable geometry contact surface C_{def}}
$\text{BVH}_{\text{def}}^{(t)} \leftarrow \texttt{UpdateBVH}\left(\text{BVH}_{\text{def}}^{(t-\Delta t)}, C_{\text{def}}^{(t)}\right)$ {Update def. geometry BVH with Algorithm 1}

$\mathcal{C}_{dd} \leftarrow \texttt{FindCollisions}\left(C_{\text{def}}^{(t)}, \text{BVH}_{\text{def}}^{(t)}\right)$ {Find deformable geometry contacts}
$F_c \leftarrow \sum_c^{\mathcal{C}_{dd}} \texttt{ComputeForces}(c, C_{\text{def}}^{(t)})$
$F_c \leftarrow \texttt{ConsolidateForces}(F_c, \mathcal{C}_{dd})$ {Correct for redundant constraints}
$R \leftarrow R + F_c$

$C_{\text{movRig}}^{(t)} \leftarrow \texttt{UpdateContactSurface}\left(C_{\text{movRig}}^{t-\Delta t}, t\right)$ {Update moving rigid geometry $C_{\text{movRig}}^{(t)}$}
$\text{BVH}_{\text{movRig}}^{(t)} \leftarrow \texttt{UpdateBVH}\left(\text{BVH}_{\text{movRig}}^{t-\Delta t}, C_{\text{movRig}}^{(t)}\right)$ {Update moving rigid-body BVHs}
$\mathcal{C}_{dr} \leftarrow \texttt{FindCollisions}\left(C_{\text{def}}^{(t)}, C_{\text{rig}}, C_{\text{movRig}}^{(t)}, \text{BVH}_{\text{def}}^{(t)}, \text{BVH}_{\text{rig}}, \text{BVH}_{\text{movRig}}^{(t)}\right)$ {Find deformable-rigid contacts}
$F_c \leftarrow \sum_c^{\mathcal{C}_{dr}} \texttt{ComputeForces}(c, C_{\text{def}}^{(t)})$
$F_c \leftarrow \texttt{ConsolidateForces}(F_c, \mathcal{C}_{dr})$
$R \leftarrow R + F_c$

$U^{t+\Delta t} \leftarrow \texttt{UpdateDisplacements}(U^{(t)}, F, R)$ {Compute next time-step displacement}

Appendix B: Derivation of the NBC computation formula

The quantities appearing in this derivation are illustrated in Fig. 2. The starting point for the derivation of our NBC formula (4) is the realisation that an optimal parent axis $\boldsymbol{a_p}$ does not only depend on the child-NBC axes, $\boldsymbol{a_1}$, $\boldsymbol{a_2}$, but also their opening angles, α_1, α_2. If β denotes the angle between the two child axes and β_1, β_2 their respective angles between them and the parent axis, the ideal opening angle of the parent NBC α_p satisfies:

$$\alpha_p = 2(\beta_1 + \alpha_1/2) = 2(\beta_2 + \alpha_2/2) \tag{23}$$

Using this and the definition of β_1, β_2, it is obtained

$$\begin{aligned} \beta &= \beta_1 + \beta_2 = \arccos(\boldsymbol{a_2}^{\mathrm{T}}\boldsymbol{a_1}) \\ \beta_1 &= (2\beta - (\alpha_1 - \alpha_2))/4, \quad \beta_2 = \beta - \beta_1 \end{aligned} \tag{24}$$

That in turn is used to obtain the desired weights w_1, w_2 for computation of the parent axis from the child axes:

$$\begin{aligned} w_1 &= \cos(\beta_1) - e\frac{\cos(\beta_2) - e\cos(\beta_1)}{1-e^2}, \quad \text{where } e := \boldsymbol{a_1}^{\mathrm{T}}\boldsymbol{a_2} \\ w_2 &= \frac{\cos(\beta_2) - e\cos(\beta_1)}{1-e^2} \\ \boldsymbol{a_p} &= \frac{1}{||w_1\boldsymbol{a_1} + w_2\boldsymbol{a_2}||}(w_1\boldsymbol{a_1} + w_2\boldsymbol{a_2}) \end{aligned} \tag{25}$$

Appendix C: Derivation of the non-rigid deformation threshold for BVH updating

A bound on the non-rigid deformation is required in the BVH updating algorithm to ensure the detection of all self-collision candidates, i.e. if no self-collision was possible in a BVH subtree at the time of its last update t_U, meaning the root node's NBC opening angle α_{VMT} is smaller than the self-collision candidate threshold τ_{VMT}, how much non-rigid deformation can the bounded geometry undergo without pushing α_{VMT} above the threshold τ_{VMT} and thus necessitating a full update the BVH subtree and check for self-collisions?

The bound introduced here, can be derived with a 2D sketch (Fig. 14): Assuming the maximum non-rigid displacement $\boldsymbol{u'}_{\text{NR,max}}$ of the bounded nodes is known, taking the smallest primitive diameter $h_{\min}$ in the set of primitives bounded by the UN, it is found that the biggest change to the primitive's normal (and thus potentially all NBCs in which it is contained) occurs if the primitives vertices both move by $||\boldsymbol{u'}_{\text{NR,max}}||$ in opposite directions and perpendicularly to the plane of the primitive. The angular change to the normal $\Delta\alpha$ arising from such non-rigid deformation is

$$\Delta\alpha = \arctan\frac{||\boldsymbol{u'}_{\text{NR,max}}||}{h_{\min}/2} \tag{26}$$

The next step is to solve for $||\boldsymbol{u'}_{\text{NR,max}}||$ by setting $\Delta\alpha$ to $(\tau_{\text{VMT}} - \alpha_{\text{VMT}}^{(t_U)})/2$, i.e. determine a safe upper bound for nodal

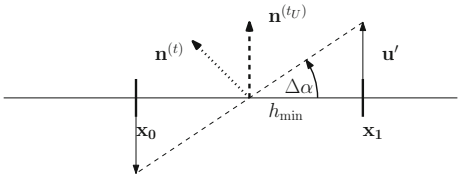


Fig. 14 In 2D: Situation leading to the largest possible change to the primitive normal for a non-rigid deformation of known magnitude $||\boldsymbol{u}'||$. The nodes of the example primitive have moved in opposite directions perpendicular to the primitive's previous plane

displacement below which self-collisions can be excluded. This immediately yields the formula:

$$||\boldsymbol{u}'_{\mathrm{NR,max}}|| = \frac{h_{\min}}{2} \tan \frac{\tau_{\mathrm{VMT}} - \alpha_{\mathrm{VMT}}^{(t_U)}}{2} \tag{27}$$

References

1. Allard J, Cotin S, Faure F, Bensoussan PJ, Poyer F, Duriez C, Delingette H, Grisoni L (2007) SOFA-an open source framework for medical simulation. Stud Health Technol Inf 125:13–18
2. Allard J, Faure F, Courtecuisse H, Falipou F, Duriez C, Kry PG (2010) Volume contact constraints at arbitrary resolution. ACM Trans Gr 29(4):1–10. doi:10.1145/1778765.1778819
3. Baumhauer M, Feuerstein M, Meinzer HP, Rassweiler J (2008) Navigation in endoscopic soft tissue surgery: perspectives and limitations. J Endourol Endourol Soc 22(4):751–766
4. Belytschko T, Neal MO (1991) Contact-impact by the pinball algorithm with penalty and Lagrangian methods. Int J Numer Methods Eng 31(3):547–572
5. Cash DM, Miga MI, Glasgow SC, Dawant BM, Clements LW, Cao Z, Galloway RL, Chapman WC (2007) Concepts and preliminary data toward the realization of image-guided liver surgery. J Gastrointest Surg Off J Soc Surg Aliment Tract 11(7):844–859
6. Cirak F, West M (2005) Decomposition contact response (DCR) for explicit finite element dynamics. Int J Numer Methods Eng 64(8):1078–1110
7. Duriez C, Dubois F, Kheddar A, Andriot C (2006) Realistic haptic rendering of interacting deformable objects in virtual environments. IEEE Trans Vis Comput Gr 12(1):36–47
8. Flores FG, Oñate E (2005) Improvements in the membrane behaviour of the three node rotation-free BST shell triangle using an assumed strain approach. Comput Methods Appl Mech Eng 194(6–8):907–932
9. Gottschalk S, Lin MC, Manocha D (1996) OBBTree: a hierarchical structure for rapid interference detection. In: Proceedings of the 23rd annual conference on computer graphics and interactive techniques. ACM, pp 171–180
10. Gupta K, Pobil AP (1998) Practical motion planning in robotics: current approaches and future directions. Wiley, London
11. Hallquist JO (2006) LS-DYNA theory manual. Technical report. Livermore Software Technology Corporation, Livermore, CA
12. Han L, Hipwell J, Taylor ZA, Tanner C, Ourselin S, Hawkes D (2010) Fast deformation simulation of breasts using GPU-based dynamic explicit finite element method. Digit Mammogr, pp 728–735
13. Heidelberger B, Teschner M, Gross M (2004) Detection of collisions and self-collisions using image-space techniques. J WSCG 12(3):145–152
14. Heinstein M (2000) Contact-impact modeling in explicit transient dynamics. Comput Methods Appl Mech Eng 187(3–4):621–640
15. Heinstein MW, Laursen TA (1999) An algorithm for the matrix-free solution of quasistatic frictional contact problems. Int J Numer Methods Eng 44(9):1205–1226
16. Hu Y, Ahmed HU, Taylor Z, Allen C, Emberton M, Hawkes D, Barratt D (2012) MR to ultrasound registration for image-guided prostate interventions. Med Image Anal 16(3):687–703
17. Hu Y, Carter TJ, Ahmed HU, Emberton M, Allen C, Hawkes DJ, Barratt DC (2011) Modelling prostate motion for data fusion during image-guided interventions. IEEE Trans Med Imaging 30(11):1887–1900
18. Johnsen SF, Taylor ZA, Clarkson M, Thompson S, Hu M, Gurusamy K, Davidson B, Hawkes DJ, Ourselin S (2012) Explicit contact modeling for surgical computer guidance and simulation. In: Holmes III DR, Wong KH (eds) Proceedings of SPIE 8316, medical imaging 2012: image-guided procedures, robotic interventions, and modeling, pp 831623-1–831623-9
19. Johnsen SF, Taylor ZA, Clarkson MJ, Hipwell J, Modat M, Eiben B, Han L, Hu Y, Mertzanidou T, Hawkes DJ, Ourselin S (2014) NiftySim: a GPU-based nonlinear finite element package for simulation of soft-tissue biomechanics. Int J Comput Assist Radiol Surg. doi:10.1007/s11548-014-1118-5
20. Kay TL, Kajiya JT (1986) Ray tracing complex scenes. In: ACM SIGGRAPH computer graphics, vol 20. ACM, pp 269–278
21. Larsson T, Akenine-Möller T (2001) Collision detection for continuously deforming bodies. Eurographics 2001:325–333
22. Larsson T, Akenine-Möller T (2006) A dynamic bounding volume hierarchy for generalized collision detection. Comput Gr 30(3):450–459
23. Lee B (2007) Physically based modelling for topology modification and deformation in surgical simulation. Ph.D. thesis, University of Sydney
24. Lin MC, Gottschalk S (1998) Collision detection between geometric models: a survey. In: Proceedings of IMA conference on mathematics of surfaces, pp 37–56
25. Mezger J, Kimmerle S, Etzmuß O (2003) Hierarchical techniques in collision detection for cloth animation. J WSCG 11(2):322–329
26. Miller K, Joldes G, Lance D (2007) Total Lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation. Commun Numer Methods Biomed Eng 23(2):121–134
27. Möller T, Trumbore B (2005) Fast, minimum storage ray/triangle intersection. In: ACM SIGGRAPH 2005 Courses. ACM, p 7
28. Otaduy MA, Tamstorf R, Steinemann D, Gross M (2009) Implicit contact handling for deformable objects. Comput Gr Forum 28(2):559–568
29. Provot X (1997) Collision and self-collision handling in cloth model dedicated to design garments. In: Graphics interface, volume 97, pp 177–189 Citeseer
30. Puso M (2004) A mortar segment-to-segment frictional contact method for large deformations. Comput Methods Appl Mech Eng 193(45–47):4891–4913
31. Simo JC (1985) A finite strain beam formulation. The three-dimensional dynamic problem. Comput Methods Appl Mech Eng 49:55–70
32. Székely G, Brechbühler C, Hutter R, Rhomberg A, Ironmonger N, Schmid P (1998) Modelling of soft tissue deformation for laparoscopic surgery simulation. In: Medical image computing and computer-assisted interventation—MICCAI '98, 1496(1):550–561
33. Taylor LM, Flanagan DP (1989) PRONTO 3D: a three-dimensional transient solid dynamics program. Technical Report SAND87-1912. Sandia National Laboratories, Albuquerque, NM

34. Taylor ZA, Cheng M, Ourselin S (2008) High-speed nonlinear finite element analysis for surgical simulation using graphics processing units. IEEE Trans Med Imaging 27(5):650–663
35. Taylor ZA, Comas O, Cheng M, Passenger J, Hawkes DJ, Atkinson D, Ourselin S (2009) On modelling of anisotropic viscoelasticity for soft tissue simulation: numerical solution and GPU execution. Med Image Anal 13(2):234–244
36. Volino P, Thalmann NM (1994) Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. Comput Gr Forum 13(3):155–166
37. Wriggers P (2002) Computational contact mechanics. Wiley, London
38. Wriggers P, Krstulovič-Opara L (2000) On Smooth Finite Element Discretizations for Frictional Contact Problems. ZAMM J Appl Math Mech 80(S1):77–80
39. Yang B, Laursen TA (2006) A contact searching algorithm including bounding volume trees applied to finite sliding mortar formulations. Comput Mech 41(2):189–205